



Mobile and Pervasive Computing 23/24

Project Final Report – BrightBalance+



Authors:	64783	André Correia
	64813	Rodrigo Fontinha
	68759	Tiago Martinho

June 7th, 2024

1. Introduction

The idea behind this project originated from the human necessity to have a conscious and respectful attitude towards the use of everyday resources, which has become more prevalent over the years. Nowadays, environments are littered with devices capturing and storing more and more information in digital format, and sometimes that becomes excessive and expensive to maintain.

BrightBalance+ is an iteration of a previous project – BrightBalance – that never came to real fruition. It consists of a smart system that focuses on light management in an attempt to automate light emission and intensity in a given household. The idea is to take the presence or absence of natural sunlight to, respectively, turn the lights off or on, raise or lower window blinds and so on. Although the main purpose of the system is to manage lighting, BrightBalance+ seeks to monitor and react to temperature as well.

2. General Overview

A server is required to store the necessary user information and the data responsible for the system's correct behavior. The solution may be applied to the multiple rooms of any building containing windows and an air conditioning system, and is to be used by various users. This implies that each room has its own microcontroller with which users interact using a mobile application, constituting a **N** clients to **M** services relationship. Each microcontroller requires two photoresistors and two thermistors.

The mobile application allows users to create their own profile to interact with the service. System specific user preferences, or "Moods" as designated in the application, are stored in the webserver and define custom light colors to allow users to apply their own twist to the service. Moods facilitate management across different rooms, as it is possible for users to create a representation of the building's blueprint in the application. Speaking of rooms, each one includes an on-demand system on or off toggle, and displays information about the temperature. Additionally, push notifications are delivered whenever the ubiquitous component suggests the raising and lowering of blinds, turning lights on and off, or modifying the emitted light color, ultimately providing control to the user.

3. System Architecture

The architecture is split into three different components, which interact with each other via WiFi (IEEE 802.11 protocol):

- **Mobile application**
Allows the communication between users and the service, programmed using Flutter with Android Studio as the IDE;
- **Ubiquitous application**
The combination of a microcontroller with sensors, actuators and a program, specialized to sensing the environment for phenomena that trigger context-aware actions. We use the Arduino IDE to control all hardware components;

➤ **Webserver**

Database server to store and provide convenient access to system and user data. The webserver of choice, Firebase, will serve as the central point of BrightBalance+, storing user and sensor data, through the Firestore database, and communicating with both the mobile and ubiquitous applications when needed.

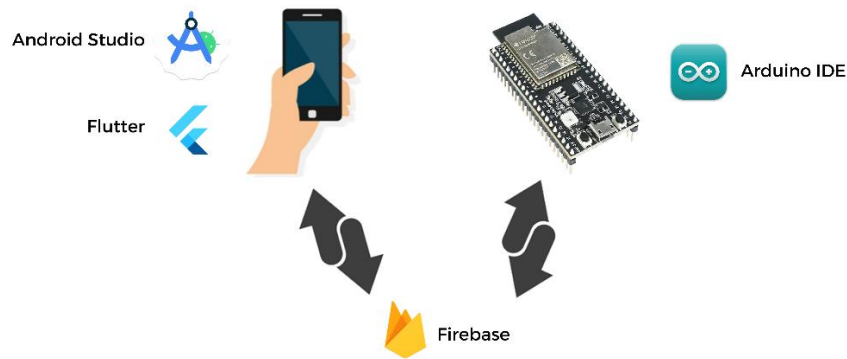


Figure 1 – System Architecture

4. Mobile Application

➤ **Create Account**

The user can create their own account by providing an email and a password.

➤ **Login Account**

The user can login into the account with their authentication credentials.

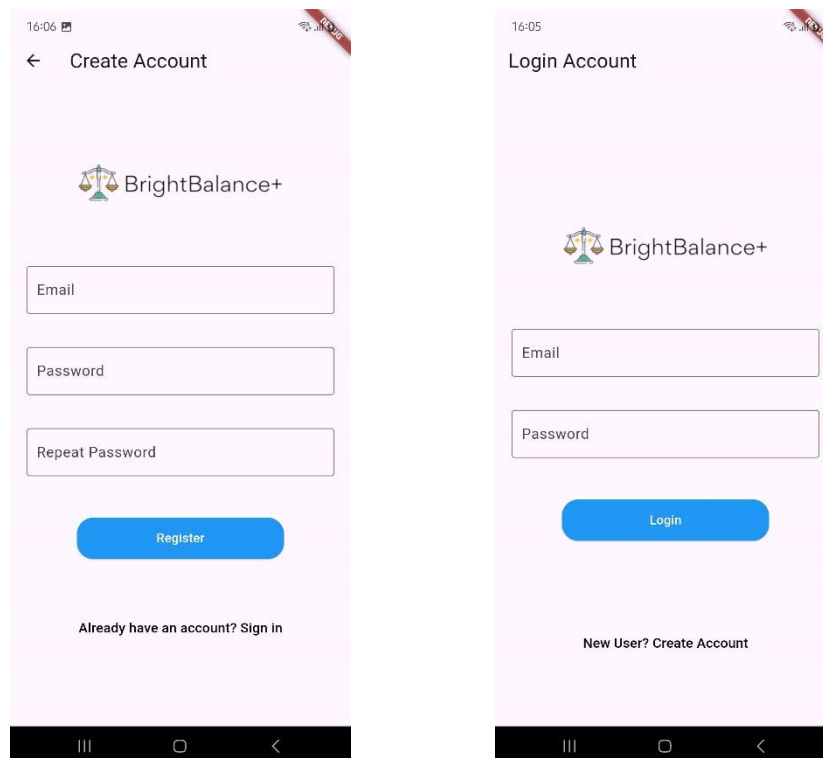


Figure 2 – Create and Login Account

➤ **Profiles**

Each account can have multiple profiles (similarly to Netflix's profile system), with each one being associated with a person, supposedly living in the same household, who wants to use the app.

➤ **Create Profile**

Anyone in the house can create their own profile, provided they can log in to the corresponding account. After logging in, the user just needs to click the "Add Profile" option, then provide a name, choose a profile picture and click "Create".

➤ **Edit Profile**

Anyone in the house can also update their profile information (name and picture). To edit a profile, users must tap a profile while the checkbox "Edit Profiles" is ticked.

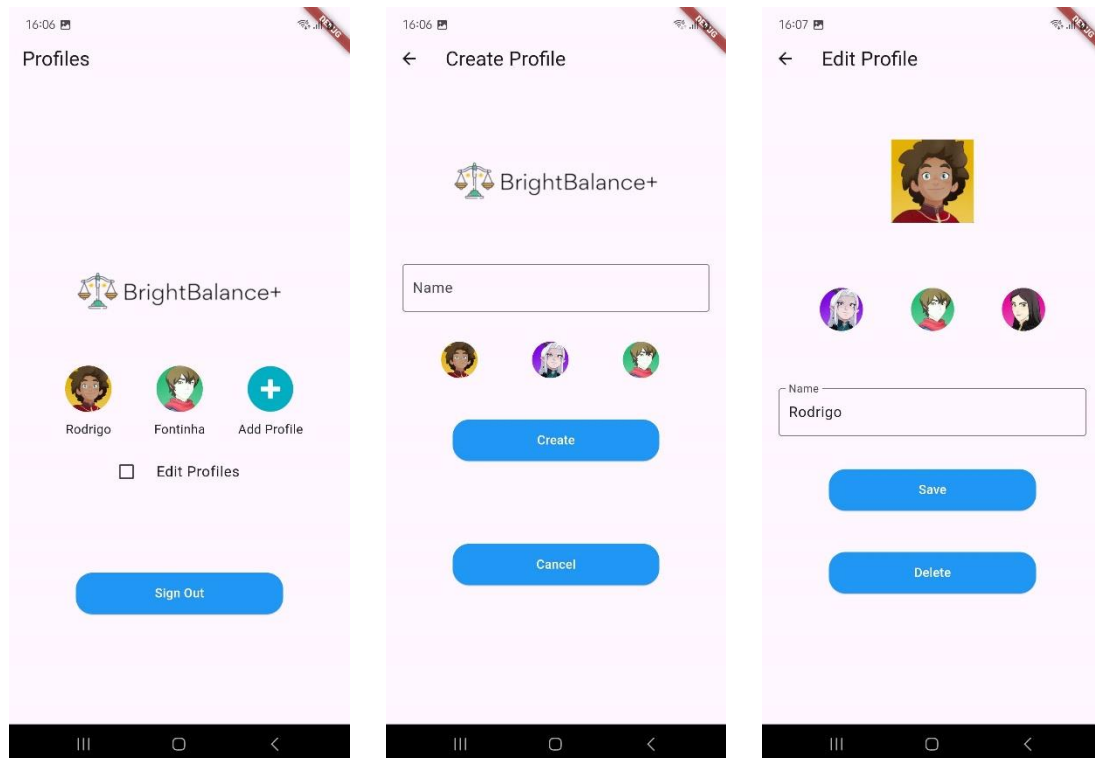


Figure 3 – Account Profiles and Create / Edit

➤ **Rooms**

When tapping a profile, the rooms of the house are shown (if created), with every profile in the same account being able to see the same set of rooms. For monitoring purposes, the temperature for a room is checked every 30 seconds, and is displayed beneath its icon, above its name.

➤ **Create Room**

Anyone in the house with a profile can create rooms by tapping "Add Room". Then, the user just needs to name the room, choose a picture for it and tap "Create".

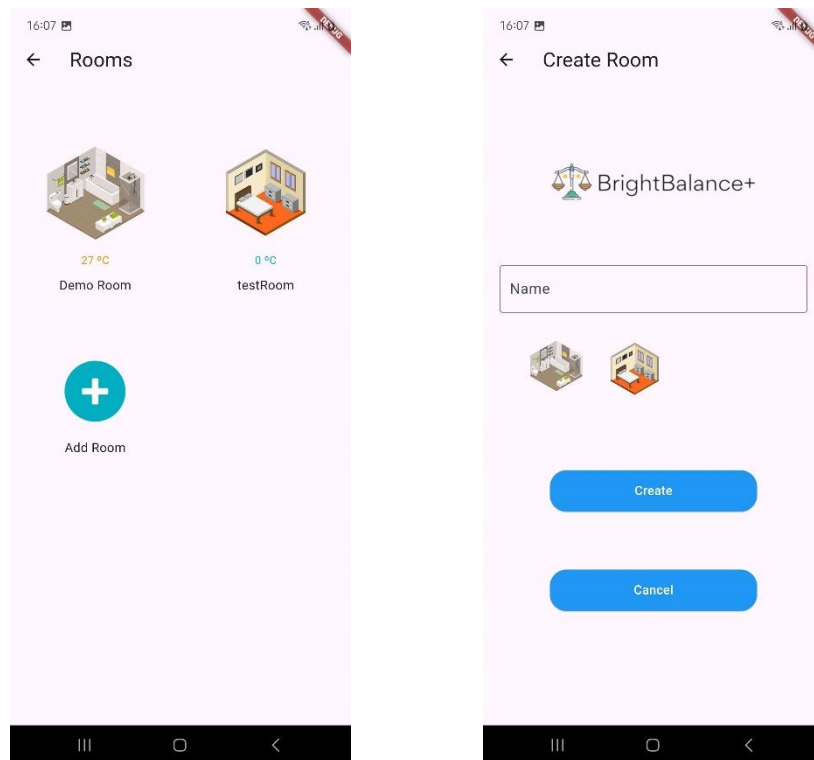


Figure 4 – Profile Rooms and Create Room

➤ Preferences / Moods

By selecting a room, a default Mood is displayed (if no other is created), and it is here where the user can activate or deactivate Moods, with the switch above the room's name. The user can change the preferences of a chosen Mood (dropdown menu) to change the color of the light in the room, adjust the position of the window blinds, and define whether they want to receive notifications regarding the fan (turn on or off) and the blinds (raise or lower). When selecting "Add Mood" on the dropdown, a popup requesting the name of the new Mood appears, and the Mood will be created by clicking "OK".

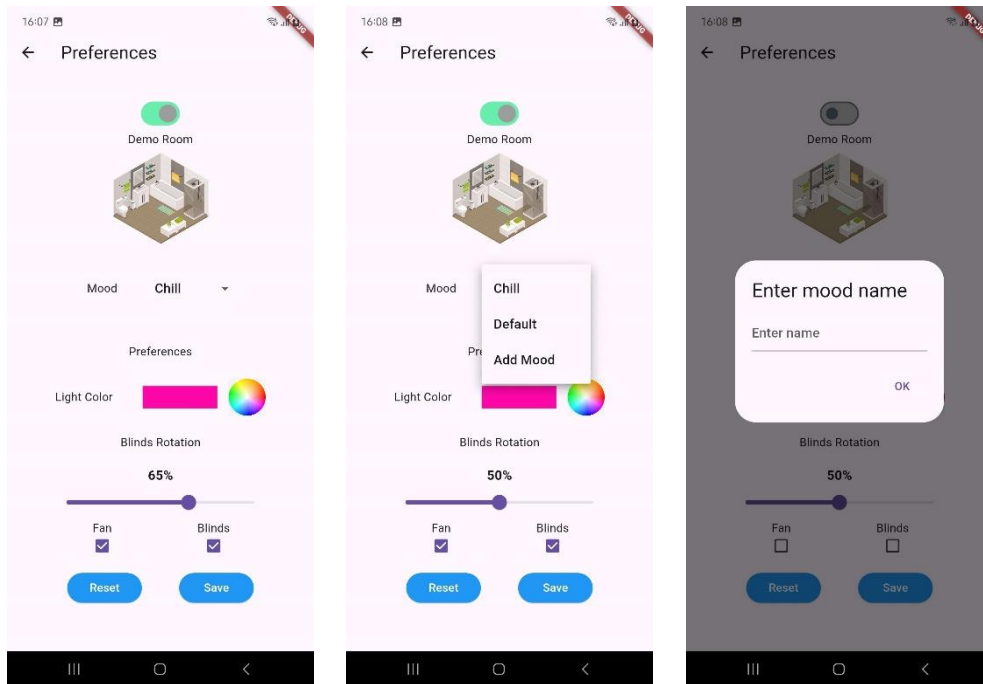


Figure 5 – Moods and Create

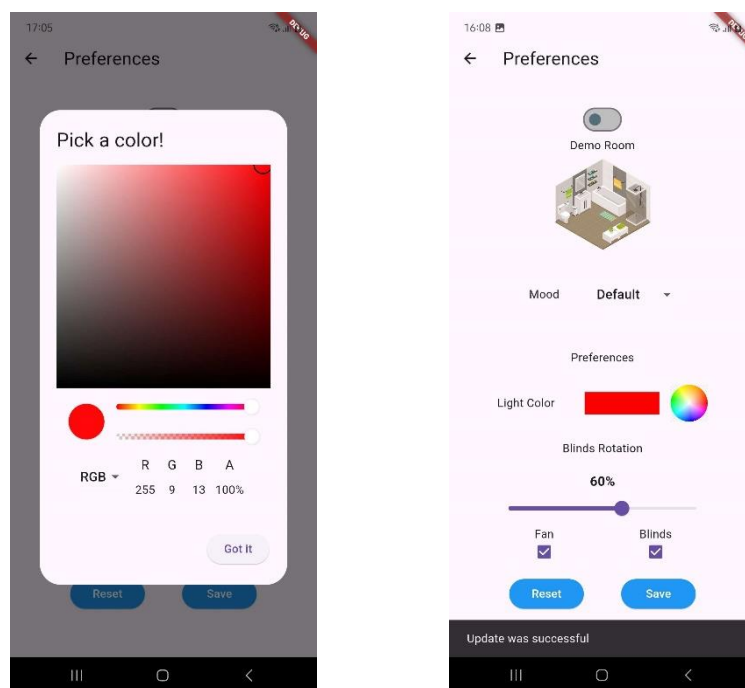


Figure 6 – Change color and update with success

➤ Notifications

For the notifications, Flutter's flutter local notifications package was used. This feature is not perfectly implemented since notifications are sent to everyone with the app open at the time of notification delivery, but the team considered this to be acceptable for proof of concept. If the "fan" and/or "blinds" checkboxes are ticked, sensor readings' changes may cause context-aware notifications to be delivered to the user. Some extra validations are also made (e.g., if the blinds are already open or closed, if the fan is already turned on or off).

For example, the first notification shown in the Figure 7 is triggered if:

- Indoor temperature is below 25 degrees Celsius;
- Checkbox for “fan” in the Mood is checked;
- The physical representation of the fan is turned on.

The second notification shown in the Figure 7 will be triggered when:

- The outdoor temperature is above 28 degrees Celsius;
- The difference between the outdoor temperature and the indoor temperature is equal or greater than 2 degrees Celsius;
- The physical representation of the blinds is not closed;

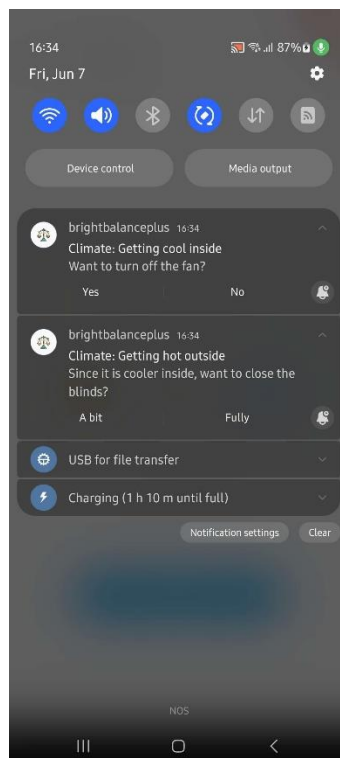


Figure 7 – Examples of notifications received based on values read by the sensors

5. Sensing and Reacting

BrightBalance+ requires a microcontroller to interact with the environment of the user's household. The module of choice is be the same ESP32 (see Figure 8) used initially for BrightBalance.



Figure 8 – WiFi module, Espressif ESP32 S2 WROOM

Table 1 includes the crucial sensors for the correct behavior of the system, particularly in collecting information about its surrounding environments. These sensors serve a purpose in capturing data to be sent to the database. The photoresistor located indoors is the only exception to this in the sense that its readings are used to calculate the brightness with which light is to be emitted.

The data gathered from photoresistors and thermistors complements one another to enhance the system's behavior. The decision to use two thermistors instead of one was based on personal experiences. In Portugal, there are many buildings with no insulation which, when combined with other factors, makes divisions reach bothersome temperatures. During the summer, when sunlight beams through windows, the incoming heat gets trapped in the building interior, causing a greenhouse effect that increases indoor temperature. In contrast, during winter times, poorly insulated buildings lose heat which decreases indoor temperature. While BrightBalance+ does not offer a solution for indoor heat loss, the idea behind having two thermistors is to be able to sense indoor and outdoor temperatures simultaneously. In case a room starts reaching temperatures hotter than that of the outside, the system can suggest raising the blinds.

In the checkpoint report, the second photoresistor was deemed unnecessary and, consequently, ended up being omitted from that version of the system. This decision was reverted because the team realized a photoresistor on the outside of a building is required to identify scenarios where the window blinds may be raised or lowered. For example, if not much natural light is sensed, nighttime may be approaching, hence the system may lower the blinds.



Sensor	Role	Image
2x LDR (Light Dependent Resistor) Photoresistors	Comparing interior with exterior light level	
2x NTC (Negative Temperature Coefficient) Thermistors	Comparing interior with exterior temperature	

Table 1 – BrightBalance+ Sensors

The system's actuators can be found in Table 2. The RGB LED is responsible for emitting light based on the current Mood (if active) with its color values, and the adequate brightness. The blinds are either raised or lowered, also based on the current Mood's setting. Lastly, the fan can be turned on in response to high temperature readings, or off if the opposite holds true. It is worthy of note that the behavior of BrightBalance+ is not entirely autonomous because user

control should come first. This is achieved through the context-aware notification system, which allows users to instruct the system’s actuators on-demand.

The prototype does not include an actual fan or blinds. Instead, the ubiquitous system uses a red LED to simulate the fan being turned on and off, and a servo motor to simulate the act of raising and lowering of the blinds.




Actuator	Role	Image
RGB LED (Light Emitter Diode)	Light emission based on environment conditions and active Mood	
Blinds	Room ambience based on sensor readings	
Fan (or other air conditioning system)	Room conditioning in case a room is too hot	

Table 2 – BrightBalance+ Actuators

The schema shown in Figure 9 represents the ubiquitous system but is not entirely faithful in regards to the prototype because the editor used for the sketch does not include all elements representative of the hardware materials used in the system. A more accurate depiction of the system is shown in Figure 10. The sensors located to the left of the RGB correspond to the outdoors, while the ones on its right represent the indoors.

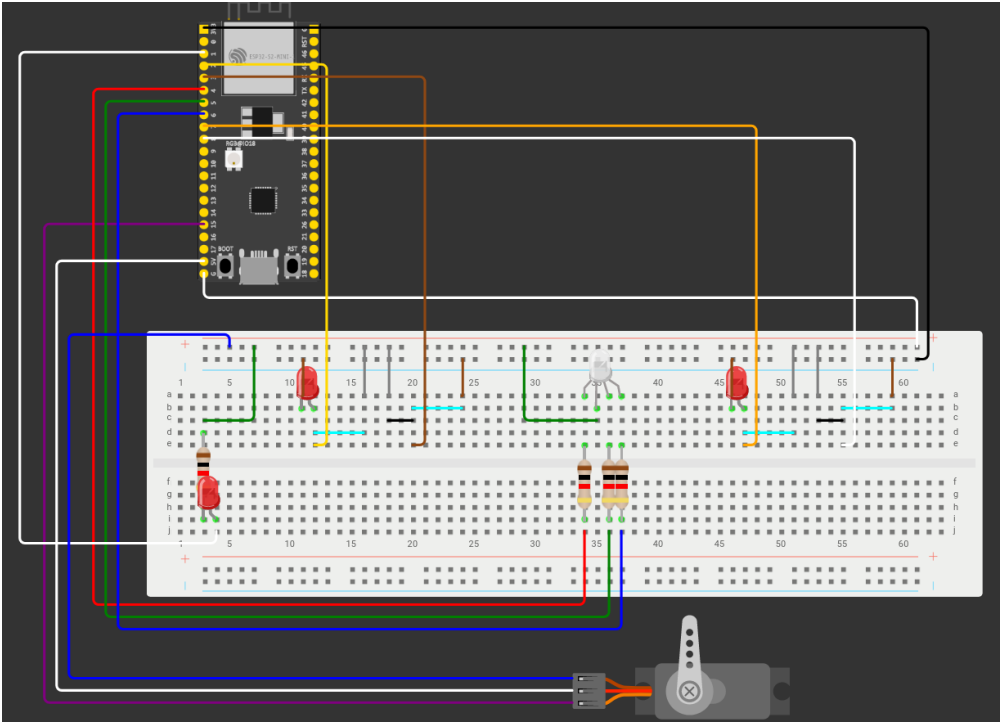


Figure 9 – Ubiquitous system simulation

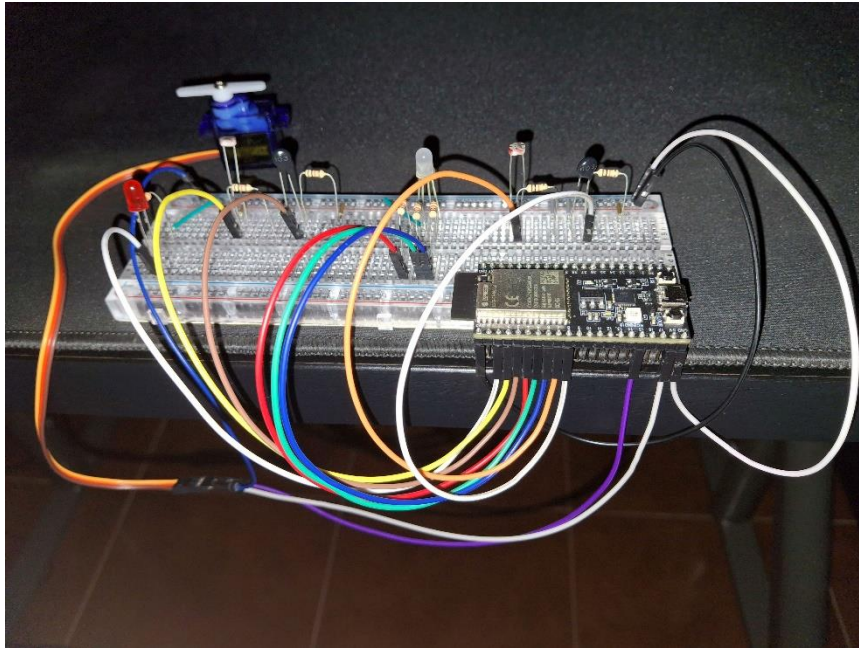


Figure 10 – Ubiquitous system

6. Experiments

To assess the correctness and usability of all of BrightBalance+ components, debugging and testing took place in various points of development of both the mobile and ubiquitous applications. Before the final delivery of the project, the team conducted three progressive experiments to ensure the validity of the solution:

1. Build the prototype and connect it to a power source (i.e., laptop) via USB. When the RGB turns on, point a flashlight over the photoresistor corresponding to the indoors (the one on the right side of the prototype in Figure 10) and slowly drag the flashlight away.

Purpose:

- Verify that a WiFi connection is correctly established;
- Check sensor readings' storage in the Firestore database;
- Ensure the RGB emits a brighter light in proportion to the absence of light indoors.

Assumptions:

- The hotspot specified in the *.ino* code file is turned on;
- Firebase is available (i.e., no service disruption or outage incidents occur).

2. Open the mobile application, log in to a pre-registered account and select a profile. Tap the "Demo Room" to then modify the preferred color in one of the Moods. Save the changes and activate the Mood.

Purpose:

- Ensure that Firestore database updates the light color upon saving the Mood;
- Ensure that Firestore database also updates the light color to be emitted by the RGB device upon activating the Mood;
- Watch the RGB changing color in response to the experiment.

Assumptions:

- Experiment 1 was completed successfully;
- Any upcoming notifications are ignored, or their suggestions denied.

- 3.** With the previous Mood, change the blinds-related slider to 65% and the color to a shade of pink, then save the changes. Await the upcoming notifications and respond to them affirmatively, one by one.

Purpose:

- Assess the servo motor simulating the lowering of the blinds, and the RGB changing color once again, upon saving the Mood changes;
- Ensure that push notifications are received and provide suggestions according to lighting and climate conditions;
- Verify that the system's devices react to user instructions, triggered by response to notification suggestions.

Assumptions:

- Experiment 2 was completed successfully;
- Ideal climate conditions: Outdoors temperature is high (equal to or above 30 degrees Celsius) and the fan is turned off;
- Ideal lighting conditions: The blinds are opened while low levels of natural light (from the outdoors) are detected.

BrightBalance+ may trigger more push notifications than expected for Experiment 3 due to natural variations in sensor readings caused by environmental phenomena. In this experiment, notifications are expected for actions to shut the blinds (rotating the servo motor to zero degrees) when it gets dark outside and to turn on the fan (simulated by the red LED) when a high temperature is measured.

7. Conclusions and Lessons Learned

During development, the BrightBalance+ team quickly came to the conclusion that some of the initially planned features would have to be reduced or even left out of the delivered version. Three examples of this are:

- Color preferences – The introduction of PWM channels to write RGB color values digitally (allowing more color variety) led to issues with the rest of the hardware components and had to be cut, so color emission is a bit limited;
- Intensity preferences – Mapping preferred minimum and maximum light intensity/brightness was too complex to justify its implementation considering the emitted light's brightness is constrained to only a handful of states;

- Mood conflicts – Handling conflicts between Moods, when two or more users are located in the same room, was omitted from the delivered version.

Hardware properties, characteristics and limitations played a substantial role in the process of handling sensor data correctly. The data sensed by the system seems to be inconsistent, as the team could not figure out how much compensation to add to the temperature readings in particular, in order for those to be more realistic. Distinguishing between day and night times is also not very accurate because all lighting-related tests were made with the help of a flashlight rather than natural light.

In a real-world implementation of the project, some functionalities would need to be added and others would require revisiting:

- The mobile application UI is to be reworked to fit a more conventional look;
- There must be a way of identifying how many users are connected to the service, and tracking them between the different rooms of the building (potentially through WiFi connection and proximity to network access points);
- Room management must be implemented;
- Each room must be assigned a unique microcontroller and check the corresponding interior temperature readings;
- Regarding Mood conflicts, user priority was considered but not implemented perfectly, so future work should address conflicts from multiple users setting different Moods for the same room.
- It would be interesting to provide monitoring as a graphical representation for a 24 or 48-hour history of light and temperature readings;
- Data gathered by sensors must be revisited.

Despite the adversities, the team feels as though the project was developed with some degree of success and ultimately helped in understanding and applying the concepts of the course.